Advanced Card Systems Ltd.
Card & Reader Technologies

ACR38x

Smart Card Reader

CCID PC/SC Memory Card Access

Subject to change without prior notice

info@acs.com.hk
www.acs.com.hk

Table of Contents

1.0. Introduction

ACR38x PC-Linked Reader acts as an interface for the communication between a computer and a smart card. Different types of smart cards have different commands and different communication protocols that in most cases, prevents a direct communication between a smart card and a computer. ACR38x reader establishes a uniform interface from the computer to the smart card for a wide variety of cards. By taking care of the card's specific particulars, it releases the computer software programmer from getting involved with the technical details of the smart card operation, which are in many cases, not relevant in the implementation of a smart card system.

This document contains the PC/SC Memory Card Command set for ACR38x (CCID).

2.0. Supported Memory Cards

ACR38x works with several memory-based smart cards such as:

- Cards following the I2Cbus protocol (free memory cards) with maximum of 128 bytes page with capability, including:

  Atmel: AT24C01/02/04/08/16/32/64/128/256/512/1024
  SGS-Thomson: ST14C02C, ST14C04C
  Gemplus: GFM1K, GFM2K, GFM4K, GFM8K

- Cards with secure memory IC with password and authentication, including:

  Atmel: AT88SC153 and AT88SC1608

- Cards with intelligent 1k bytes EEPROM with write-protect function, including:

  Infineon: SLE4418, SLE4428, SLE5518 and SLE5528

- Cards with intelligent 256 bytes EEPROM with write-protect function, including:

  Infineon: SLE4432, SLE4442, SLE5532 and SLE5542

- Cards with '104' type EEPROM non-reloadable token counter cards, including:

  Infineon: SLE4406, SLE4436, SLE5536 and SLE6636

- Cards with Intelligent 416-Bit EEPROM with internal PIN check, including:

  Infineon: SLE4404

- Cards with Security Logic with Application Zone(s), including:

  Atmel: AT88SC101, AT88SC102 and AT88SC1003

3.0. Memory Card Type Selection

3.1. By Programmatic Method

SELECT_CARD_TYPE command must be executed first before other memory card commands. This command powers down and powers up the selected card that is inserted in the card reader and performs a card reset. This command can only be used after the logical smart card reader communication has been established using the *SCardConnect( )* API. For details of *SCardConnect( )* API, please refer to PC/SC specifications. For the **Memory Card Command Set**, please refer to Section 4.0.

A code snippet for the program flow is given below to demonstrate how to select the memory card type in ACR38x (CCID):

```
SCARDCONTEXT hContext;
SCARDHANDLE hCard;
unsigned long dwActProtocol;
SCARD_IO_REQUEST ioRequest;
DWORD size = 64, SendLen = 6, RecvLen = 255, retCode;
byte cardType;

//Establish PC/SC Connection
retCode = SCardEstablishContext (SCARD_SCOPE_USER, NULL, NULL,
&hContext);

//List all readers in the system
retCode = SCardListReaders (hContext, NULL, readerName, &size);

//Connect to the reader
retCode = SCardConnect(hContext, readerName, SCARD_SHARE_SHARED,
SCARD_PROTOCOL_T0, &hCard, &dwActProtocol);

//Select Card Type
unsigned char SendBuff[] = {0xFF,0xA4,0x00,0x00,0x01,cardType};
retCode = SCardTransmit( hCard, &ioRequest, SendBuff, SendLen, NULL,
RecvBuff, &RecvLen);

//Disconnect from the reader
retCode = SCardDisconnect(hCard, SCARD_UNPOWER_CARD);

//End the established context
retCode = SCardReleaseContext(hContext);
```

4.0. Memory Card Command Set

This section contains the Memory Card Command Set for ACR38x (CCID).

4.1. Recollection Card – 1, 2, 4, 8 and 18 Kbit I2C Card

4.1.1. SELECT_CARD_TYPE

This command powers down and powers up the selected card that is inserted in the card reader and performs a card reset.

> **Note:** This command can only be used after the logical smart card reader communication has been established using the *SCardConnect( )* API. For details of *SCardConnect( )* API, please refer to PC/SC specifications.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FF$_H$	A4$_H$	00$_H$	00$_H$	01$_H$	01$_H$

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

4.1.2. SELECT_PAGE_SIZE

This command chooses the page size to read the smart card. The default value is 8-byte page write. It will reset to default value whenever the card is removed or the reader is powered off.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Page Size
FFH	01H	00H	00H	01H	

Page size	= 03H for 8-byte page write
	= 04H for 16-byte page write
	= 05H for 32-byte page write
	= 06H for 64-byte page write
	= 07H for 128-byte page write

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2	= 90H 00H if no error

4.1.3. READ_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FFH	B0H			

Byte Address — Memory address location of the memory card

Byte Address Memory address location of the memory card

MEM_L Length of data to be read from the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

BYTE 1	…	…	BYTE N	SW1	SW2

BYTE x Data read from memory card

SW1, SW2 = 90H 00H if no error

4.1.4. WRITE_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	Byte Address		MEM_L	Byte 1	....	....	Byte n
		MSB	LSB					
FF_H	D0_H							

Byte Address Memory address location of the memory card

MEM_L Length of data to be written to the memory card

Byte x Data to be written to the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

4.2. Memory Card – 32, 64, 128, 256, 512, and 1024 Kbit I2C Card

4.2.1. SELECT_CARD_TYPE

This command powers down and powers up the selected card that is inserted in the card reader and performs a card reset.

> **Note:** This command can only be used after the logical smart card reader communication has been established using the *SCardConnect( )* API. For details of *SCardConnect( )* API, please refer to PC/SC specifications.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FF_H	$A4_H$	00_H	00_H	01_H	02_H

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

4.2.2. SELECT_PAGE_SIZE

This command chooses the page size to read the smart card. The default value is eight-byte page write. It will reset to default value whenever the card is removed or the reader is powered off.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Page size
FF_H	01_H	00_H	00_H	01_H	

Data	TPDU to be sent to the card
Page size	= 03_H for 8-byte page write
	= 04_H for 16-byte page write
	= 05_H for 32-byte page write
	= 06_H for 64-byte page write
	= 07_H for 128-byte page write

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2	= 90_H 00_H if no error

4.2.3. READ_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FF_H				

INS = $B0_H$ for 32, 64, 128, 256, 512kbit iic card

= $1011\ 000{*}_b$ for 1024kbit iic card,

where * is the MSB of the 17 bit addressing

Byte Address Memory address location of the memory card

MEM_L Length of data to be read from the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

BYTE 1	…	…	BYTE N	SW1	SW2

BYTE x Data read from memory card

SW1, SW2 = 90_H 00_H if no error

4.2.4. WRITE_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	Byte Address		MEM_L	Byte 1	....	....	Byte n
		MSB	LSB					
FF$_H$								

INS = D0$_H$ for 32, 64, 128, 256, 512kbit iic card

= 1101 000*$_b$ for 1024kbit iic card,

where * is the MSB of the 17 bit addressing

Byte Address Memory address location of the memory card

MEM_L Length of data to be written to the memory card

Byte x Data to be written to the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

4.3. Memory Card – ATMEL AT88SC153

4.3.1. SELECT_CARD_TYPE

This command powers down and powers up the selected card that is inserted in the card reader and performs a card reset. It will also select the page size to be 8-byte page write.

> **Note:** This command can only be used after the logical smart card reader communication has been established using the *SCardConnect( )* API. For details of *SCardConnect( )* API, please refer to PC/SC specifications.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FFH	A4H	00H	00H	01H	03H

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90H 00H if no error

4.3.2. READ_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FF~H~		00~H~		

INS = B0~H~ for reading zone 00~b~

= B1~H~ for reading zone 01~b~

= B2~H~ for reading zone 10~b~

= B3~H~ for reading zone 11~b~

= B4~H~ for reading fuse

Byte Address Memory address location of the memory card

MEM_L Length of data to be read from the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

BYTE 1	…	…	BYTE N	SW1	SW2

BYTE x Data read from memory card

SW1, SW2 = 90~H~ 00~H~ if no error

4.3.3. WRITE_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	P1	Byte Address	MEM_L	Byte 1	....	....	Byte n
FF$_H$		00$_H$						

INS = D0$_H$ for writing zone 00$_b$

= D1$_H$ for writing zone 01$_b$

= D2$_H$ for writing zone 10$_b$

= D3$_H$ for writing zone 11$_b$

= D4$_H$ for writing fuse

Byte Address Memory address location of the memory card

MEM_L Length of data to be written to the memory card

MEM_D Data to be written to the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

4.3.4. VERIFY_PASSWORD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU							
CLA	INS	P1	P2	Lc	Pw(0)	Pw(1)	Pw(2)
FF_H	20_H	00_H		03_H			

Pw(0),Pw(1),Pw(2) Passwords to be sent to memory card

P2 = 0000 00rp$_b$

where the two bits “rp” indicate the password to compare

r = 0: Write password,

r = 1: Read password,

p : Password set number,

rp = 01 for the secure code.

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2 ErrorCnt
90$_H$	

SW1 = 90$_H$

SW2 (ErrorCnt) = Error Counter. FF$_H$ indicates the verification is correct. 00$_H$ indicates the password is locked (or exceeded the maximum number of retries). Other values indicate the current verification has failed.

4.3.5. INITIALIZE_AUTHENTICATION

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	P1	P2	Lc	Q(0)	Q(1)	…	Q(7)
FF_H	84_H	00_H	00_H	08_H				

Q(0),Q(1)…Q(7) Host random number, 8 bytes

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

4.3.6. VERIFY_AUTHENTICATION

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	P1	P2	Lc	Ch(0)	Ch(1)	…	Ch(7)
FF$_H$	82$_H$	00$_H$	00$_H$	08$_H$				

Ch(0),Ch(1)…Ch(7) Host challenge, 8 bytes

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

4.4. Memory Card – ATMEL AT88C1608

4.4.1. SELECT_CARD_TYPE

This command powers down and powers up the selected card that is inserted in the card reader and performs a card reset. It will also select the page size to be 16-byte page write.

> **Note:** This command can only be used after the logical smart card reader communication has been established using the *SCardConnect( )* API. For details of *SCardConnect( )* API, please refer to PC/SC specifications.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FF_H	$A4_H$	00_H	00_H	01_H	04_H

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

4.4.2. READ_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	Zone Address	Byte Address	MEM_L
FF$_H$				

INS = B0$_H$ for reading user zone

= B1$_H$ for reading configuration zone or reading fuse

Zone Address = 0000 0$A_{10}A_9A_8$ $_b$ where A_{10} is the MSB of zone address

= don't care for reading fuse

Byte Address = $A_7A_6A_5A_4$ $A_3A_2A_1A_0$ $_b$ is the memory address location of the memory card

= 1000 0000$_b$ for reading fuse

MEM_L Length of data to be read from the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

BYTE 1	…	…	BYTE N	SW1	SW2

BYTE x Data read from memory card

SW1, SW2 = 90$_H$ 00$_H$ if no error

4.4.3. WRITE_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	Zone Address	Byte Address	MEM_L	Byte 1	…	…	Byte n
FFH								

INS = D0H for writing user zone

= D1H for writing configuration zone or writing fuse

Zone Address = 0000 0$A_{10}A_9A_8$ b where A_{10} is the MSB of zone address

= Don't care for writing fuse

Byte Address = $A_7A_6A_5A_4$ $A_3A_2A_1A_0$ b is the memory address location of the memory card

= 1000 0000b for writing fuse

MEM_L Length of data to be written to the memory card

Byte x Data to be written to the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90H 00H if no error

4.4.4. VERIFY_PASSWORD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	P1	P2	Lc	Data			
FFH	20H	00H	00H	04H	RP	Pw(0)	Pw(1)	Pw(2)

Pw(0),Pw(1),Pw(2) Passwords to be sent to memory card

RP = 0000 $rp_2p_1p_0$ $_b$

where the four bits "$rp_2p_1p_0$" indicate the password to compare:

r = 0 : Write password,

r = 1 : Read password,

$p_2p_1p_0$: Password set number.

($rp_2p_1p_0$ = 0111 for the secure code)

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2 ErrorCnt
90H	

SW1 = 90H

SW2 (ErrorCnt) = Error Counter. FFH indicates the verification is correct. 00H indicates the password is locked (or exceeded the maximum number of retries). Other values indicate the current verification has failed.

4.4.5. INITIALIZE_AUTHENTICATION

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	P1	P2	Lc	Q(0)	Q(1)	…	Q(7)
FF~H~	84~H~	00~H~	00~H~	08~H~				

Byte Address	Memory address location of the memory card
Q(0),Q(1)…Q(7)	Host random number, 8 bytes

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90~H~ 00~H~ if no error

4.4.6. VERIFY_AUTHENTICATION

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	P1	P2	Lc	Q1(0)	Q1(1)	…	Q1(7)
FF_H	82H	00H	00H	08H				

Byte Address Memory address location of the memory card

Q1(0),Q1(1)…Q1(7) Host challenge, 8 bytes

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90H 00H if no error

4.5. Memory Card – SLE 4418 / SLE 4428 / SLE 5518 / SLE 5528

4.5.1. SELECT_CARD_TYPE

This command powers down and powers up the selected card that is inserted in the card reader and performs a card reset.

> **Note:** This command can only be used after the logical smart card reader communication has been established using the *SCardConnect( )* API. For details of *SCardConnect( )* API, please refer to PC/SC specifications.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FF_H	$A4_H$	00_H	00_H	01_H	05_H

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

4.5.2. READ_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FF_H	$B0_H$			

MSB Byte Address = $0000\ 00A_9A_8\ _b$ is the memory address location of the memory card

LSB Byte Address = $A_7A_6A_5A_4\ \ A_3A_2A_1A_0\ \ _b$ is the memory address location of the memory card

MEM_L Length of data to be read from the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

BYTE 1	…	…	BYTE N	SW1	SW2

BYTE x Data read from memory card

SW1, SW2 = $90_H\ 00_H$ if no error

4.5.3. READ_PRESENTATION_ERROR_COUNTER_MEMORY_CARD (SLE 4428 and SLE 5528)

This command is used to read the presentation error counter for the secret code.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	P1	P2	MEM_L
FF_H	B1H	00H	00H	03H

Response Data Format (abData field in the RDR_to_PC_DataBlock)

ERRCNT	DUMMY 1	DUMMY 2	SW1	SW2

ERRCNT Error Counter. FFH indicates that the last verification is correct. 00H indicates that the password is locked (exceeded the maximum number of retries). Other values indicate that the last verification has failed.

DUMMY Two bytes dummy data read from the card

SW1, SW2 = 90H 00H if no error

4.5.4. READ_PROTECTION_BIT

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FF_H	$B2_H$			

MSB Byte Address = $0000\ 00A_9A_8$ b is the memory address location of the memory card

LSB Byte Address = $A_7A_6A_5A_4\ \ A_3A_2A_1A_0$ b is the memory address location of the memory card

MEM_L Length of protection bits to be read from the card, in multiples of 8 bits. Maximum value is 32.

MEM_L = 1 + INT((number of bits - 1) /8)

For example, to read eight protection bits starting from memory 0x0010, the following pseudo-APDU should be issued:

```
0xFF 0xB2 0x00 0x10 0x01
```

Response Data Format (abData field in the RDR_to_PC_DataBlock)

PROT 1	…	…	PROT L	SW1	SW2

PROT y Bytes containing the protection bits

SW1, SW2 = 90_H 00_H if no error

The arrangement of the protection bits in the PROT bytes is as follows:

PROT 1								PROT 2								…							
P8	P7	P6	P5	P4	P3	P2	P1	P16	P15	P14	P13	P12	P11	P10	P9	..	..	..	..	..	..	P18	P17

Px is the protection bit of BYTE x in the response data

'0' byte is write protected

'1' byte can be written

4.5.5. WRITE_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	Byte Address		MEM_L	Byte 1	....	....	Byte N
		MSB	LSB					
FF H	D0 H							

MSB Byte Address	= 0000 00A_9A_8 b is the memory address location of the memory card
LSB Byte Address	= $A_7A_6A_5A_4$ $A_3A_2A_1A_0$ b is the memory address location of the memory card
MEM_L	Length of data to be written to the memory card
Byte x	Data to be written to the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

4.5.6. WRITE_PROTECTION_MEMORY_CARD

Each byte specified in the command is used in the card to compare the byte stored in a specified address location. If the data match, the corresponding protection bit is irreversibly programmed to ‘0’.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	Byte Address		MEM_L	Byte 1	....	....	Byte N
		MSB	LSB					
FF$_H$	D1$_H$							

MSB Byte Address = 0000 00A_9A_8 $_b$ is the memory address location of the memory card

LSB Byte Address = $A_7A_6A_5A_4$ $A_3A_2A_1A_0$ $_b$ is the memory address location of the memory card

MEM_L Length of data to be written to the memory card

Byte x Byte values to be compared with the data in the card starting at *Byte Address*. BYTE 1 is compared with the data at Byte Address; BYTE N is compared with the data at (Byte Address+N-1).

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

4.5.7. PRESENT_CODE_MEMORY_CARD (SLE 4428 and SLE 5528)

This command is used to submit the secret code to the memory card to enable the write operation with the SLE 4428 and SLE 5528 card, the following actions are executed:

1. Search a '1' bit in the presentation error counter and write the bit to '0'
2. Present the specified code to the card
3. Try to erase the presentation error counter

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU						
CLA	INS	P1	P2	MEM_L	CODE	
					Byte 1	Byte 2
FFH	20H	00H	00H	02H		

CODE Two bytes secret code (PIN)

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2 ErrorCnt
90H	

SW1 = 90H

SW2 (ErrorCnt) = Error Counter. FFH indicates successful verification. 00H indicates that the password is locked (or exceeded the maximum number of retries). Other values indicate that current verification has failed.

4.6. Memory Card – SLE 4432 / SLE 4442 / SLE 5532 / SLE 5542

4.6.1. SELECT_CARD_TYPE

This command powers down and powers up the selected card that is inserted in the card reader and performs a card reset.

> **Note:** This command can only be used after the logical smart card reader communication has been established using the *SCardConnect( )* API. For details of *SCardConnect( )* API, please refer to PC/SC specifications.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FF~H~	A4~H~	00~H~	00~H~	01~H~	06~H~

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90~H~ 00~H~ if no error

4.6.2. READ_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FFH	B0H	00H		

Byte Address = $A_7A_6A_5A_4\ A_3A_2A_1A_0$ b is the memory address location of the memory card

MEM_L Length of data to be read from the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

BYTE 1	…	…	BYTE N	SW1	SW2

BYTE x Data read from memory card

SW1, SW2 = 90H 00H if no error

4.6.3. READ_PRESENTATION_ERROR_COUNTER_MEMORY_CARD (SLE 4442 and SLE 5542)

This command is used to read the presentation error counter for the secret code.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	P1	P2	MEM_L
FF_H	$B1_H$	00_H	00_H	04_H

Response Data Format (abData field in the RDR_to_PC_DataBlock)

ERRCNT	DUMMY 1	DUMMY 2	DUMMY 3	SW1	SW2

ERRCNT Error counter. 07_H indicates that the last verification is correct. 00_H indicates that the password is locked (exceeded the maximum number of retries). Other values indicate that the last verification has failed.

DUMMY Three bytes dummy data read from the card

SW1, SW2 = 90_H 00_H if no error

4.6.4. READ_PROTECTION_BITS

To read the protection bits for the first 32 bytes.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	P1	P2	MEM_L
FF$_H$	B2$_H$	00$_H$	00$_H$	04$_H$

Response Data Format (abData field in the RDR_to_PC_DataBlock)

PROT 1	PROT 2	PROT 3	PROT 4	SW1	SW2

PROT y Bytes containing the protection bits from protection memory

SW1, SW2 = 90$_H$ 00$_H$ if no error

The arrangement of the protection bits in the PROT bytes is as follows:

PROT 1								PROT 2								...							
P8	P7	P6	P5	P4	P3	P2	P1	P16	P15	P14	P13	P12	P11	P10	P9	..	..	..	..	..	..	P18	P17

Px is the protection bit of BYTE x in the response data

'0' byte is write protected

'1' byte can be written

4.6.5. WRITE_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	P1	Byte Address	MEM_L	Byte 1	....	....	Byte N
FF_H	$D0_H$	00_H						

Byte Address = $A_7A_6A_5A_4\ A_3A_2A_1A_0\ _b$ is the memory address location of the memory card

MEM_L Length of data to be written to the memory card

Byte x Data to be written to the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

4.6.6. WRITE_PROTECTION_MEMORY_CARD

Each byte specified in the command is internally in the card compared with the byte stored at the specified address and if the data match, the corresponding protection bit is irreversibly programmed to ‘0’.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	P1	Byte Address	MEM_L	Byte 1	....	....	Byte N
FF_H	$D1_H$	00_H						

Byte Address = $000A_4\ A_3A_2A_1A_0\ _b$ (00_H to $1F_H$) is the protection memory address location of the memorycard

MEM_L Length of data to be written to the memorycard

Byte x Byte values to be compared with the data in the card starting at Byte Address. BYTE 1 is compared with the data at Byte Address; BYTE N is compared with the data at (Byte Address+N-1).

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = $90_H\ 00_H$ if no error

4.6.7. PRESENT_CODE_MEMORY_CARD (SLE 4442 and SLE 5542)

To submit the secret code to the memory card to enable the write operation with the SLE 4442 and SLE 5542 card, the following actions are executed:

1. Search a '1' bit in the presentation error counter and write the bit to '0'
2. Present the specified code to the card
3. Try to erase the presentation error counter

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU							
CLA	INS	P1	P2	MEM_L	CODE		
					Byte 1	Byte 2	Byte 3
FFH	20H	00H	00H	03H			

CODE Three bytes secret code (PIN)

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2 ErrorCnt
90H	

SW1 = 90H

SW2 (ErrorCnt) = Error Counter. 07H indicates that the verification is correct. 00H indicates the password is locked (exceeded the maximum number of retries). Other values indicate that the current verification has failed.

4.6.8. CHANGE_CODE_MEMORY_CARD (SLE 4442 and SLE 5542)

This command is used to write the specified data as new secret code in the card.

The current secret code must have been presented to the card with the *PRESENT_CODE* command prior to the execution of this command.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU							
CLA	INS	P1	P2	MEM_L	CODE		
					Byte 1	Byte 2	Byte 3
FF$_H$	D2$_H$	00$_H$	01$_H$	03$_H$			

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

4.7. Memory Card – SLE 4406 / SLE 4436 / SLE 5536 / SLE 6636

4.7.1. SELECT_CARD_TYPE

This command powers down and powers up the selected card that is inserted in the card reader and performs a card reset.

> **Note:** This command can only be used after the logical smart card reader communication has been established using the *SCardConnect( )* API. For details of *SCardConnect( )* API, please refer to PC/SC specifications.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FF_H	$A4_H$	00_H	00_H	01_H	07_H

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

4.7.2. READ_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FF~H~	B0~H~	00~H~		

Byte Address = Memory address location of the memory card

MEM_L Length of data to be read from the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

BYTE 1	…	…	BYTE N	SW1	SW2

BYTE x Data read from memory card

SW1, SW2 = 90~H~ 00~H~ if no error

4.7.3. WRITE_ONE_BYTE_MEMORY_CARD

This command is used to write one byte to the specified address of the inserted card. The byte is written to the card with LSB first, i.e., the bit at card address 0 is regarded as the LSB of byte 0.

Four different WRITE modes are available for this card type, which are distinguished by a flag in the command data field:

a) ***Write***

The byte value specified in the command is written to the specified address. This command can be used for writing personalization data and counter values to the card.

b) ***Write with carry***

The byte value specified in the command is written to the specified address and the command is sent to the card to erase the next lower counter stage. Thus, this write mode can only be used for updating the counter value in the card.

c) ***Write with backup enabled*** (SLE 4436, SLE 5536 and SLE 6636 only)

The byte value specified in the command is written to the specified address. This command can be used for writing personalization data and counter values to the card. Backup bit is enabled to prevent data loss when card tearing occurs.

d) ***Write with carry and backup enabled*** (SLE 4436, SLE 5536 and SLE 6636 only)

The byte value specified in the command is written to the specified address and the command is sent to the card to erase the next lower counter stage. Thus, this write mode can only be used for updating the counter value in the card. Backup bit is enabled to prevent data loss when card tearing occurs.

With all write modes, the byte at the specified card address is not erased prior to the write operation and, hence, memory bits can only be programmed from '1' to '0'.

The backup mode available in the SLE 4436 and SLE 5536 card can be enabled or disabled in the write operation.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU						
CLA	INS	P1	Byte Address	MEM_L	MODE	BYTE
FF$_H$	D0$_H$	00$_H$		02$_H$		

Byte Address = Memory address location of the memory card

MODE Specifies the write mode and backup option

00$_H$: Write

01$_H$: Write with carry

02$_H$: Write with backup enabled (SLE 4436, SLE 5536 and SLE 6636 only)

03$_H$: Write with carry and with backup enabled (SLE 4436, SLE 5536 and SLE 6636 only)

BYTE Byte value to be written to the card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

4.7.4. PRESENT_CODE_MEMORY_CARD

To submit the secret code to the memory card to enable the card personalization mode, the following actions are executed:

1. Search a '1' bit in the presentation counter and write the bit to '0'
2. Present the specified code to the card

The ACR38x does not try to erase the presentation counter after the code submission. This must be done by the application software through a separate ‘Write with carry' command.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	P1	P2	MEM_L	CODE			
					ADDR	Byte 1	Byte 2	Byte 3
FF$_H$	20$_H$	00$_H$	00$_H$	04$_H$	09$_H$			

ADDR Byte address of the presentation counter in the card

CODE Three bytes secret code (PIN)

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

4.7.5. AUTHENTICATE_MEMORY_CARD (SLE 4436, SLE 5536 and SLE 6636)

To read a card authentication certificate from a SLE 5536 or SLE 6636 card, the ACR38x executes the following actions:

1. Select Key 1 or Key 2 in the card as specified in the command
2. Present the challenge data specified in the command to the card
3. Generate the specified number of CLK pulses for each bit of authentication data computed by the card
4. Read 16 bits of authentication data from the card
5. Reset the card to normal operation mode

The authentication has to be performed in two steps. The first step is to send the Authentication Certificate to the card. The second step is to get back two bytes of authentication data calculated by the card.

Step 1: Send Authentication Certificate to the Card

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU											
CLA	INS	P1	P2	MEM_L	CODE						
					KEY	CLK_CNT	Byte 1	Byte 2	……	Byte 5	Byte 6
FF$_H$	84$_H$	00$_H$	00$_H$	08$_H$							

KEY Key to be used for the computation of the authentication certificate:

00$_H$: Key 1 with no cipher block chaining

01$_H$: Key 2 with no cipher block chaining

80$_H$: Key 1 with cipher block chaining (SLE 5536 and SLE 6636 only)

81$_H$: Key 2 with cipher block chaining (SLE 5536 and SLE 6636 only)

CLK_CNT Number of CLK pulses to be supplied to the card for the computation of each bit of the authentication certificate. Typical value is 160 clocks (A0$_H$)

BYTE 1...6 Card challenge data

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2
61$_H$	02$_H$

SW1, SW2 = 61$_H$ 02$_H$ if no error, meaning two bytes of authentication data are ready. The authentication data can be retrieved by *Get_Response* command.

Step 2: Get back the Authentication Data (Get_Response)

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	P1	P2	MEM_L
FF$_{H}$	C0$_{H}$	00$_{H}$	00$_{H}$	02$_{H}$

Response Data Format (abData field in the RDR_to_PC_DataBlock)

CERT		SW1	SW2

CERT 16 bits of authentication data computed by the card. The LSB of BYTE 1 is the first authentication bit read from the card.

SW1, SW2 = 90$_{H}$ 00$_{H}$ if no error

4.8. Memory Card – SLE 4404

4.8.1. SELECT_CARD_TYPE

This command powers down and powers up the selected card that is inserted in the card reader and performs a card reset.

> **Note:** This command can only be used after the logical smart card reader communication has been established using the *SCardConnect( )* API. For details of *SCardConnect( )* API, please refer to PC/SC specifications.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FF$_H$	A4$_H$	00$_H$	00$_H$	01$_H$	08$_H$

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

4.8.2. READ_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FF_H	B0_H	00_H		

Byte Address = Memory address location of the memory card

MEM_L Length of data to be read from the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

BYTE 1	…	…	BYTE N	SW1	SW2

BYTE x Data read from memory card

SW1, SW2 = 90_H 00_H if no error

4.8.3. WRITE_MEMORY_CARD

This command is used to write data to the specified address of the inserted card. The byte is written to the card with LSB first, i.e., the bit at card address 0 is regarded as the LSB of byte 0.

The byte at the specified card address is not erased prior to the write operation and, hence, memory bits can only be programmed from '1' to '0'.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	P1	Byte Address	MEM_L	Byte 1	…	…	Byte N
FF$_H$	D0$_H$	00$_H$						

Byte Address = Memory address location of the memory card

MEM_L Length of data to be written to the memory card

BYTE Byte value to be written to the card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

4.8.4. ERASE_SCRATCH_PAD_MEMORY_CARD

This command is used to erase the data of the scratch pad memory of the inserted card. All memory bits inside the scratch pad memory will be programmed to the state of '1'.

To erase error counter or user area, please use the *VERIFY_USER_CODE* command as specified in the Section 4.8.5.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FF$_H$	D2$_H$	00$_H$		00$_H$

Byte Address = Memory byte address location of the scratch pad

Typical value is 0x02

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

4.8.5. VERIFY_USER_CODE

This command is used to submit User Code (2 bytes) to the inserted card. User Code is to enable the memory access of the card.

The following actions are executed:

1. Present the specified code to the card
2. Search a '1' bit in the presentation error counter and write the bit to '0'
3. Erase the presentation error counter. The User Error Counter can be erased when the submitted code is correct.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU						
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE	
					Byte 1	Byte 2
FF_H	20_H	04_H	08_H	02_H		

Error Counter LEN	Length of presentation error counter in bits
Byte Address	Byte address of the key in the card
CODE	2 bytes User Code

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

= 63_H 00_H if there are no more retries

Note: After SW1SW2 = 0x9000 has been received, read back the User Error Counter to check if the *VERIFY_USER_CODE* is correct. If User Error Counter is erased and is equal to "0xFF", the previous verification is successful.

4.8.6. VERIFY_MEMORY_CODE

This command is used to submit Memory Code (4 bytes) to the inserted card. Memory Code is used to authorize the reloading of the user memory, together with the User Code.

The following actions are executed:

1. Present the specified code to the card
2. Search a '1' bit in the presentation error counter and write the bit to '0'
3. Erase the presentation error counter. Please note that Memory Error Counter cannot be erased.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE			
					Byte 1	Byte 2	Byte 3	Byte 4
FFH	20H	40H	28H	04H				

Error Counter LEN Length of presentation error counter in bits

Byte Address Byte address of the key in the card

CODE 4 bytes Memory Code

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

= 63_H 00_H if there are no more retries

Note: After SW1SW2 = 0x9000 has been received, read back the Application Area can check if the *VERIFY_MEMORY_CODE* is correct. If all data in Application Area is erased and is equal to “0xFF”, the previous verification is successful.

4.9. Memory Card – AT88SC101 / AT88SC102 / AT88SC1003

4.9.1. SELECT_CARD_TYPE

This command powers down and powers up the selected card that is inserted in the card reader and performs a card reset.

> **Note:** This command can only be used after the logical smart card reader communication has been established using the *SCardConnect( )* API. For details of *SCardConnect( )* API, please refer to PC/SC specifications.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FF$_H$	A4$_H$	00$_H$	00$_H$	01$_H$	09$_H$

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

4.9.2. READ_MEMORY_CARD

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FFH	B0H	00H		

Byte Address = Memory address location of the memory card

MEM_L Length of data to be read from the memory card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

BYTE 1	…	…	BYTE N	SW1	SW2

BYTE x Data read from memory card

SW1, SW2 = 90H 00H if no error

4.9.3. WRITE_MEMORY_CARD

This command is used to write data to the specified address of the inserted card. The byte is written to the card with LSB first, i.e., the bit at card address 0 is regarded as the LSB of byte 0.

The byte at the specified card address is not erased prior to the write operation and, hence, memory bits can only be programmed from '1' to '0'.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	P1	Byte Address	MEM_L	Byte 1	....	....	Byte N
FF_H	$D0_H$	00_H						

Byte Address = Memory address location of the memory card

MEM_L Length of data to be written to the memory card

BYTE Byte value to be written to the card

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

4.9.4. ERASE_NON_APPLICATION_ZONE

This command is used to erase the data in Non-Application Zones. The EEPROM memory is organized into 16-bit words. Although erases are performed on single bit, the ERASE operation clears an entire word in the memory. Therefore, performing an ERASE on any bit in the word will clear ALL 16 bits of that word to the state of '1'.

To erase Error Counter or the data in Application Zones, please refer to the following:

1. *ERASE_APPLICATION_ZONE_WITH_ERASE* command as specified in Section 4.9.5
2. *ERASE_APPLICATION_ZONE_WITH_WRITE_AND_ERASE* command as specified in Section 4.9.6
3. *VERIFY_SECURITY_CODE* commands as specified in Section 4.9.7

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FF_H	$D2_H$	00_H		00_H

Byte Address = Memory byte address location of the word to be erased.

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

4.9.5. ERASE_APPLICATION_ZONE_WITH_ERASE

This command can be used in the following cases:

1. AT88SC101: To erase the data in Application Zone with EC Function Disabled
2. AT88SC102: To erase the data in Application Zone 1
3. AT88SC102: To erase the data in Application Zone 2 with EC2 Function Disabled
4. AT88SC1003: To erase the data in Application Zone 1
5. AT88SC1003: To erase the data in Application Zone 2 with EC2 Function Disabled
6. AT88SC1003: To erase the data in Application Zone 3

The following actions are executed for this command:

1. Present the specified code to the card
2. Erase the presentation error counter. The data in corresponding Application Zone can be erased when the submitted code is correct.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU									
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE				
					Byte 1	Byte 2	…	…	Byte N
FF$_H$	20$_H$	00$_H$							

Error Counter LEN Length of presentation error counter in bits. The value should be 0x00 always.

Byte Address Byte address of the Application Zone Key in the card. Please refer to the table below for the correct value.

	Byte Address	LEN
AT88SC101: Erase Application Zone with EC function disabled	96$_H$	04$_H$
AT88SC102: Erase Application Zone 1	56$_H$	06$_H$
AT88SC102: Erase Application Zone 2 with EC2 function disabled	9C$_H$	04$_H$
AT88SC1003: Erase Application Zone 1	36$_H$	06$_H$
AT88SC1003: Erase Application Zone 2 with EC2 function disabled	5C$_H$	04$_H$
AT88SC1003: Erase Application Zone 3	C0$_H$	06$_H$

MEM_L Length of the Erase Key. Please refer to the table above for the correct value.

CODE N bytes of Erase Key

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

Note: After SW1SW2 = 0x9000 has been received, read back the data in Application Zone to check if the *ERASE_APPLICATION_ZONE_WITH_ERASE* is correct. If all data in Application Zone is erased and is equal to "0xFF", the previous verification is successful.

4.9.6. ERASE_APPLICATION_ZONE_WITH_WRITE_AND_ERASE

This command can be used in the following cases:

1. AT88SC101: To erase the data in Application Zone with EC Function Enabled
2. AT88SC102: To erase the data in Application Zone 2 with EC2 Function Enabled
3. AT88SC1003: To erase the data in Application Zone 2 with EC2 Function Enabled

With EC or EC2 Function Enabled (that is, ECEN or EC2EN Fuse is unblown and in "1" state), the following actions are executed:

1. Present the specified code to the card
2. Search a '1' bit in the presentation error counter and write the bit to '0'
3. Erase the presentation error counter. The data in corresponding Application Zone can be erased when the submitted code is correct.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE			
					Byte 1	Byte 2	Byte 3	Byte 4
FF_H	20_H	80_H		04_H				

Error Counter LEN Length of presentation error counter in bits. The value should be 0x80 always.

Byte Address Byte address of the Application Zone Key in the card

	Byte Address
AT88SC101	96_H
AT88SC102	9C_H
AT88SC1003	5C_H

CODE 4 bytes Erase Key

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

= 63_H 00_H if there are no more retries

Note: After SW1SW2 = 0x9000 has been received, read back the data in Application Zone can check whether the *ERASE_APPLICATION_ZONE_WITH_WRITE_AND_ERASE* is correct. If all data in Application Zone is erased and is equal to "0xFF", the previous verification is successful.

4.9.7. VERIFY_SECURITY_CODE

This command is used to submit Security Code (2 bytes) to the inserted card. Security Code is to enable the memory access of the card.

The following actions are executed:

1. Present the specified code to the card
2. Search a '1' bit in the presentation error counter and write the bit to '0'
3. Erase the presentation error counter. The Security Code Attempts Counter can be erased when the submitted code is correct.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU						
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE	
					Byte 1	Byte 2
FF$_H$	20$_H$	08$_H$	0A$_H$	02$_H$		

Error Counter LEN Length of presentation error counter in bits

Byte Address Byte address of the key in the card

CODE 2 bytes Security Code

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90$_H$ 00$_H$ if no error

= 63$_H$ 00$_H$ if there are no more retries

Note: After SW1SW2 = 0x9000 has been received, read back the Security Code Attempts Counter (SCAC) to check whether the *VERIFY_USER_CODE* is correct. If SCAC is erased and is equal to "0xFF," the previous verification is successful.

4.9.8. BLOWN_FUSE

This command is used to blow the fuse of the inserted card. The fuse can be EC_EN Fuse, EC2EN Fuse, Issuer Fuse or Manufacturer’s Fuse.

Note: The blowing of Fuse is an irreversible process.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU								
					CODE			
CLA	INS	Error Counter LEN	Byte Address	MEM_L	Fuse Bit Addr (High)	Fuse Bit Addr (Low)	State of FUS Pin	State of RST Pin
FF_H	05_H	00_H	00_H	04_H			01_H	00_H or 01_H

Fuse Bit Addr (2 bytes) Bit address of the fuse. Please refer to the table below for the correct value.

State of FUS Pin State of the FUS pin. Should always be `0x01`.

State of RST Pin State of the RST pin. Please refer to below table for the correct value.

		Fuse Bit Addr (High)	Fuse Bit Addr (Low)	State of RST Pin
AT88SC101	Manufacturer Fuse	05_H	80_H	01_H
	EC_EN Fuse	05_H	$C9_H$	01_H
	Issuer Fuse	05_H	$E0_H$	01_H
AT88SC102	Manufacturer Fuse	05_H	$B0_H$	01_H
	EC2EN Fuse	05_H	$F9_H$	01_H
	Issuer Fuse	06_H	10_H	01_H
AT88SC1003	Manufacturer Fuse	03_H	$F8_H$	00_H
	EC2EN Fuse	03_H	FC_H	00_H
	Issuer Fuse	03_H	$E0_H$	00_H

Response Data Format (abData field in the RDR_to_PC_DataBlock)

SW1	SW2

SW1, SW2 = 90_H 00_H if no error

4.10. Other Commands Access via PC_to_RDR_XfrBlock

4.10.1. GET_READER_INFORMATION

This command returns relevant information about the particular ACR38x (CCID) model and the current operating status, such as, the firmware revision number, the maximum data length of a command and response, the supported card types, and whether a card is inserted and powered up or not.

> **Note:** This command can only be used after the logical smart card reader communication has been established using the *SCardConnect( )* API. For details of *SCardConnect( )* API, please refer to PC/SC specifications.

Command Format (abData field in the PC_to_RDR_XfrBlock)

Pseudo-APDU				
CLA	INS	P1	P2	Lc
FF$_H$	09$_H$	00$_H$	00$_H$	10$_H$

Response Data Format (abData field in the RDR_to_PC_DataBlock)

FIRMWARE										MAX_C	MAX_R	C_TYPE		C_SEL	C_STAT

FIRMWARE 10 bytes data for firmware version

MAX_C The maximum number of command data bytes

MAX_R The maximum number of data bytes that can be requested to be transmitted in a response

C_TYPE The card types supported by the ACR38x (CCID). This data field is a bitmap with each bit representing a particular card type. A bit set to '1' means the corresponding card type is supported by the reader and can be selected with the *SELECT_CARD_TYPE* command. The bit assignment is as follows:

Byte	1								2							
card type	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

Refer to the next section for the correspondence between these bits and the respective card types.

C_SEL The currently selected card type. A value of 00H means that no card type has been selected.

C_STAT Indicates whether a card is physically inserted in the reader and whether the card is powered up:

00$_H$: No card inserted

01$_H$: Card inserted, not powered up

03$_H$: Card powered up

Appendix A. Supported Card Types

The following table summarizes the card type returned by *GET_READER_INFORMATION* correspond with the respective card type.

Byte	Card Type
00H	Auto-select T=0 or T=1 communication protocol
01H	I2C memory card (1k, 2k, 4k, 8k and 16k bits)
02H	I2C memory card (32k, 64k, 128k, 256k, 512k and 1024k bits)
03H	Atmel AT88SC153 secure memory card
04H	Atmel AT88SC1608 secure memory card
05H	Infineon SLE 4418 and SLE 4428
06H	Infineon SLE 4432 and SLE 4442
07H	Infineon SLE 4406, SLE 4436 and SLE 5536
08H	Infineon SLE 4404
09H	Atmel AT88SC101, AT88SC102 and AT88SC1003
0CH	MCU-based cards with T=0 communication protocol
0DH	MCU-based cards with T=1 communication protocol

Appendix B. Response Error Codes

The following table summarizes the possible error code returned by the ACR38x (CCID):

Error Code	Status
FF$_H$	SLOTERROR_CMD_ABORTED
FE$_H$	SLOTERROR_ICC_MUTE
FD$_H$	SLOTERROR_XFR_PARITY_ERROR
FC$_H$	SLOTERROR_XFR_OVERRUN
FB$_H$	SLOTERROR_HW_ERROR
F8$_H$	SLOTERROR_BAD_ATR_TS
F7$_H$	SLOTERROR_BAD_ATR_TCK
F6$_H$	SLOTERROR_ICC_PROTOCOL_NOT_SUPPORTED
F5$_H$	SLOTERROR_ICC_CLASS_NOT_SUPPORTED
F4$_H$	SLOTERROR_PROCEDURE_BYTE_CONFLICE
F3$_H$	SLOTERROR_DEACTIVATED_PROTOCOL
F2$_H$	SLOTERROR_BUSY_WITH_AUTO_SEQUENCE
E0$_H$	SLOTERROR_CMD_SLOT_BUSY